DW_lp_cntr_updn_df

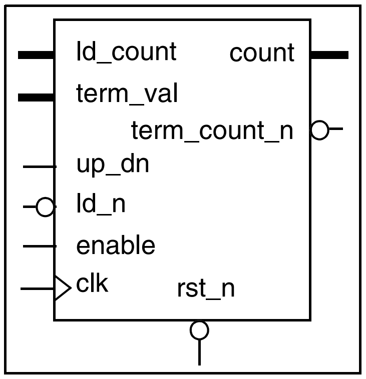
Low Power Counter with Dynamic Terminal Count Flag

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History

- Reduced power through efficient clock gating
- Parameter-controlled counter width
- Dynamically controlled terminal count flag specification
- Parameter-controlled selection of registered flag output
- Parameter-controlled selection of synchronous versus asynchronous reset
- Synchronous enable
- Synchronous load



Description

DW_lp_cntr_updn_df is a general-purpose up/down counter with dynamic count-to logic.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Clock
rst_n	1 bit	Input	Reset input (active low)
enable	1 bit	Input	Enable input
up_dn	1 bit	Input	Up versus down control input (1 = up, 0 = down)
ld_n	1 bit	Input	Counter load control (active low)
ld_count	*width* bits	Input	Counter load value
term_val	*width* bits	Input	Terminal count value
count	*width* bits	Output	Output count bus
term_count_n	1 bit	Output	Terminal count (active low)

Table 1-2 Parameter Description

Parameter	Values	Description
width	1 to 2048	Word length of counter
rst_mode	0 or 1 Default: 0	Reset mode 0 = Asynchronous reset 1 = Synchronous reset
reg_trmcnt	0 or 1 Default: 0	Register term_count_n output 0 = Not registered 1 = Registered

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	■ DesignWare (P-2019.03 and later) ■ DesignWare-LP[a] (before P-2019.03)

a. For versions before P-2019.03, you must enable minPower as follows:
`set_synthetic_library {dw_foundation.sldb dw_minpower.sldb}`

Table 1-4 Simulation Models

Model	Function
DW03.DW_LP_CNTR_UPDN_DF_CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW_lp_cntr_updn_df_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_lp_cntr_updn_df.v	Verilog simulation model source code

Table 1-5 Counter Operation Truth Table

rst_n	ld_n	enable	up_dn	Operation
0	X	X	X	Reset
1	0	X	X	Load
1	1	0	X	Hold (disabled)
1	1	1	0	Count down
1	1	1	1	Count up

When the count value equals the value on the `term_val` port, the signal `term_count_n` (terminal count) is asserted (low). The signal `term_count_n` can be connected directly to `ld_n` to synchronously reset the counter to a predefined value on the input pin of the data bus, `ld_count`.

The counter is *width* bits wide and has 2^{width} states from “000...0” to “111...1”. The counter is clocked on the positive edge of `clk`.

The reset signal, `rst_n`, is active low and provides for an asynchronous reset of the counter to “000...0”. If the reset pin is connected to ‘1’, then the reset logic is not synthesized, resulting in a smaller and faster counter.

The `term_val` is an input bus that ranges from 0 to *width* - 1. When the counter output, `count`, equals `term_val`, `term_count_n` goes active (low) for one clock cycle.

The `up_dn` input controls whether the counter counts up (`up_dn` is high) or down (`up_dn` is low), starting on the next positive edge of `clk`.

The counter is loaded with `data` by asserting `ld_n` (low) and applying data to `ld_count`. The data load operation is synchronous with respect to the positive edge of `clk`.

The count enable pin, `enable`, is active high. When `enable` is high, the counter is active. When `enable` is low, the counter is disabled, and `count` remains at the same value.

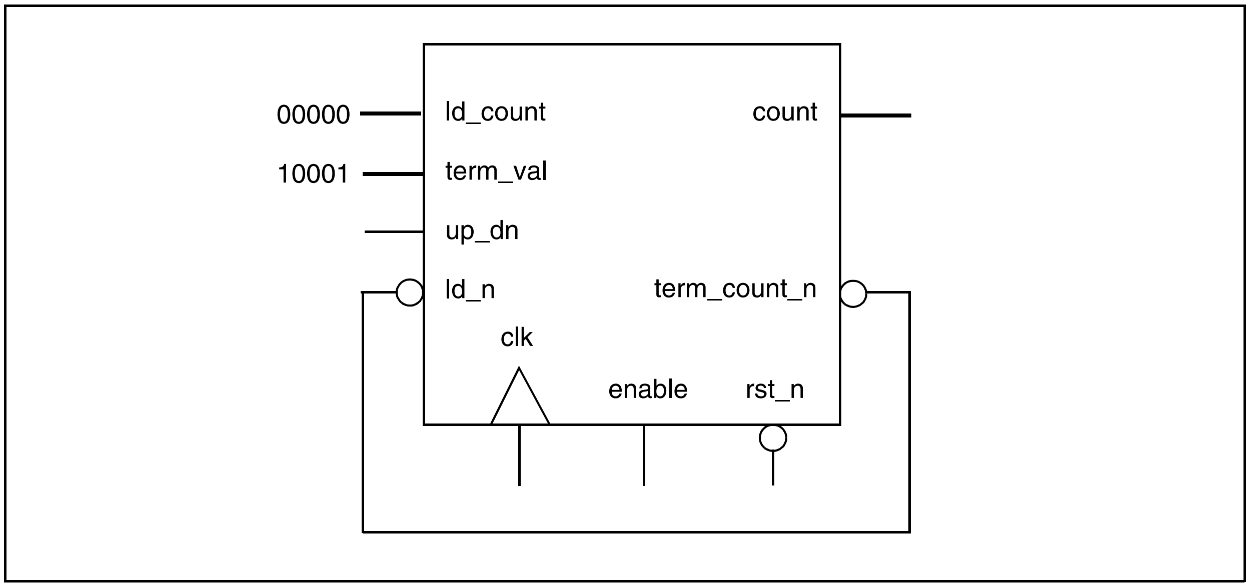
Application Example

An example application of DW_lp_cntr_updn_df is to count from 0 to 17, repeatedly. This is done by:

1. Connecting the terminal count output signal, `term_count_n`, to the `ld_n` input.
2. Setting the `ld_count` input to the start of the count sequence (for example, “00000”).
3. Connecting the `term_val` input to the end of the count sequence (for example, “10001”).

The `count` output then cycles through the states 00000 (`ld_count`) to 10001 (`term_val`); see Figure 1-1.

Figure 1-1 Counter Application: *width* = 5



Timing Diagrams

The following figures show various timing diagrams for DW_lp_cntr_updn_df.

Figure 1-2 Functional Operation: Reset, Load, and Count-to Sequence

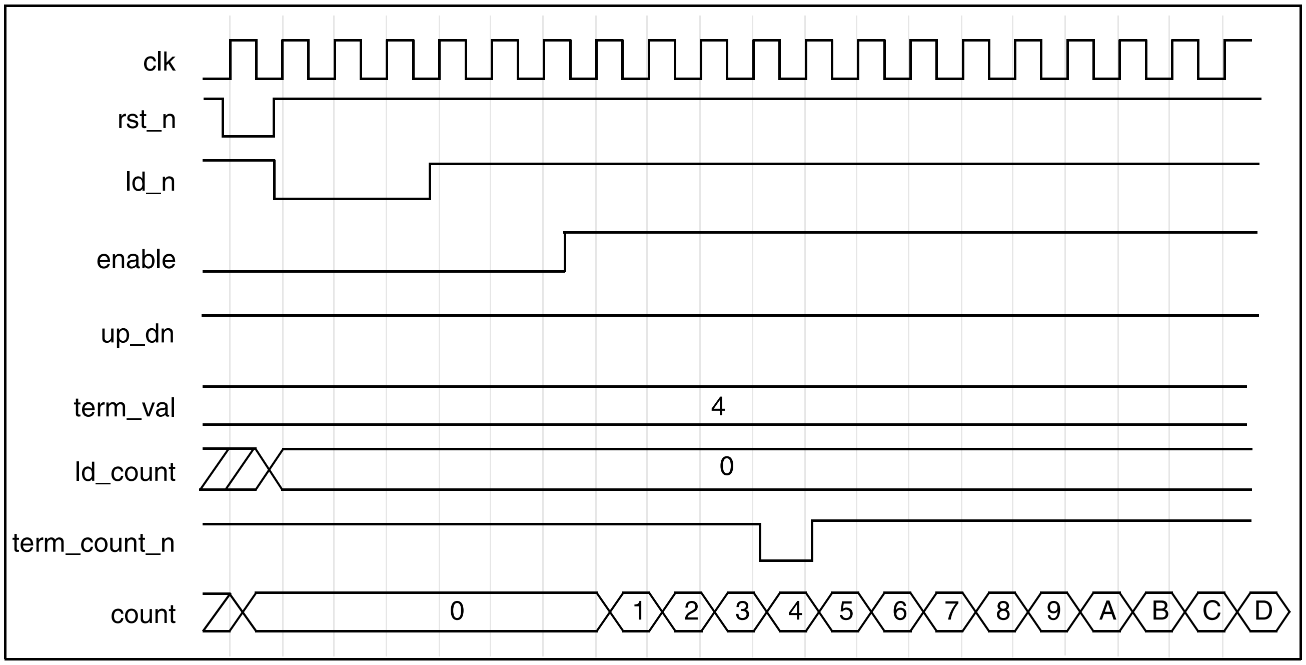
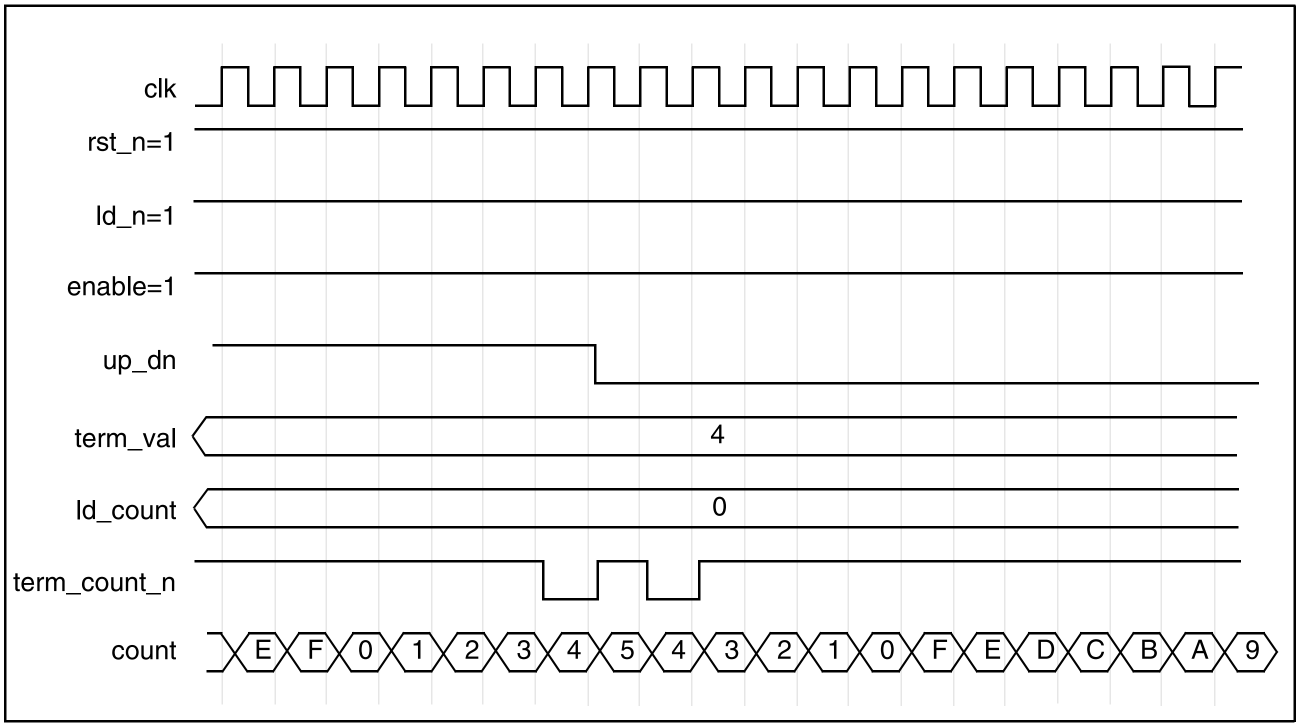


Figure 1-3 Functional Operation: Up and Down Counting, and Count-to Sequence



Related Topics

- Logic – Sequential Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_Foundation_comp.all;

entity DW_lp_cntr_updn_df_inst is
      generic (
        inst_width : POSITIVE := 8;
        inst_rst_mode : NATURAL := 0;
        inst_reg_trmcnt : NATURAL := 0
        );
      port (
        inst_clk : in std_logic;
        inst_rst_n : in std_logic;
        inst_enable : in std_logic;
        inst_up_dn : in std_logic;
        inst_ld_n : in std_logic;
        inst_ld_count : in std_logic_vector(inst_width-1 downto 0);
        inst_term_val : in std_logic_vector(inst_width-1 downto 0);
        count_inst : out std_logic_vector(inst_width-1 downto 0);
        term_count_n_inst : out std_logic
        );
    end DW_lp_cntr_updn_df_inst;


architecture inst of DW_lp_cntr_updn_df_inst is

begin

    -- Instance of DW_lp_cntr_updn_df
    U1 : DW_lp_cntr_updn_df
    generic map (width => inst_width,
                rst_mode => inst_rst_mode,
                reg_trmcnt => inst_reg_trmcnt )

    port map ( clk => inst_clk, rst_n => inst_rst_n,
             enable => inst_enable, up_dn => inst_up_dn,
             ld_n => inst_ld_n, ld_count => inst_ld_count,
             term_val => inst_term_val,
             count => count_inst, term_count_n => term_count_n_inst );

end inst;
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_lp_cntr_updn_df_inst(
          inst_clk, inst_rst_n,
          inst_enable, inst_up_dn,
          inst_ld_n, inst_ld_count,
          inst_term_val,
          count_inst, term_count_n_inst );

parameter width = 8;
parameter rst_mode = 0;
parameter reg_trmcnt = 0;


input inst_clk;
input inst_rst_n;
input inst_enable;
input inst_up_dn;
input inst_ld_n;
input [width-1 : 0] inst_ld_count;
input [width-1 : 0] inst_term_val;
output [width-1 : 0] count_inst;
output term_count_n_inst;

    // Instance of DW_lp_cntr_updn_df
    DW_lp_cntr_updn_df #(width, rst_mode, reg_trmcnt)
      U1 (.clk(inst_clk), .rst_n(inst_rst_n),
          .enable(inst_enable), .up_dn(inst_up_dn),
          .ld_n(inst_ld_n), .ld_count(inst_ld_count),
          .term_val(inst_term_val),
          .count(count_inst), .term_count_n(term_count_n_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
March 2019	DWBB_201903.0	■ Clarified some information about minPower in Table 1-3 on page 2 ■ Added this Revision History table and the document links on this page

Copyright Notice and Proprietary Information